Interactive Evolution of Complex Behaviours through Skill Encapsulation

Pablo González de Prado Salas and Sebastian Risi
{pago, sebr}@itu.dk

IT University of Copenhagen, Denmark

Abstract. Human-based computation (HBC) is an emerging research area in which humans and machines collaborate to solve tasks that neither one can solve in isolation. In evolutionary computation, HBC is often realized through interactive evolutionary computation (IEC), in which a user guides evolution by iteratively selecting the parents for the next generation. IEC has shown promise in a variety of different domains, but evolving more complex or hierarchically composed behaviours remains challenging with the traditional IEC approach. To overcome this challenge, this paper combines the recently introduced ESP (*encapsulation, syllabus and pandemonium*) algorithm with IEC to allow users to intuitively break complex challenges into smaller pieces and preserve, reuse and combine interactively evolved sub-skills. The combination of ESP principles with IEC provides a new way in which human insights can be leveraged in evolutionary computation and, as the results in this paper show, IEC-ESP is able to solve complex control problems that are challenging for a traditional fitness-based approach.

Keywords: Evolutionary computation; interactive evolutionary computation; modular networks; neuroevolution

1 Introduction

A promising approach to solve complex control problems is *neuroevolution* [1, 2], an iterative process that takes natural evolution as inspiration to train artificial neural networks (ANNs). Each generation produces new networks through mutations, and selection guides this process to create ANNs that produce specific behaviours. While neuroevolution has shown promise in a variety of tasks [1, 3], a major problem with the traditionally employed objective functions in evolutionary computation is that they can be vulnerable to deception wherein evolution converges to a suboptimal solution [4, 5]. Fitness evaluations can also be problematic for abstract tasks, for which designing an efficient evaluation function may be almost as challenging as solving the problem itself.

One way to deal with deception in evolution is to involve humans in the loop to evaluate candidate solutions, a method known as *interactive evolutionary computation* (IEC; [6]). Especially when combined with more open-ended

search methods such as novelty search [4, 7], it is possible to create synergistic effects between humans and automated processes to solve more complex tasks [8, 9]. However, most IEC systems suffer from two drawbacks. First, a significant problem is human fatigue [6], which means that users can only evaluate a limited set of candidates at a time. Second, in most existing IEC systems it is difficult to incrementally elaborate on previously evolved behaviours or to create hierarchically composed behaviours [6, 8, 9].

This paper introduces a method that aims to scale IEC approaches to more complex domains and behaviours. Our approach builds on the ESP (*encapsulation, syllabus and pandemonium*) method that allows a complex task to be broken down into intermediate learning tasks, together with mechanisms for preservation, reuse, and combination of previously learned skills [10, 11]. While the original ESP method relied on multiple pre-defined fitness functions, the new method *IEC-ESP* allows users to break down a task into smaller sub-tasks based on their human-level domain knowledge; modules for the sub-skills can then be evolved separately and protected from further evolution through encapsulation. Regulatory modules orchestrate when the different modules should be activated. Once the basic behaviours are evolved interactively, regulatory modules can be evolved interactively as well. These skills can in turn be encapsulated and used as a unit for further evolution.

While other research areas have been able to exploit a large range of human intuition in the context of human-based computation [12] (a prominent example being the Foldit game, in which users need to determine the three-dimensional structure of proteins [13]), current IEC systems limit the user to solely "nudging" evolution by deciding between a discrete choice of candidates. The new approach presented in this paper goes a step further and also allows users to use their intuition in breaking down complex behaviors into microtasks (often employed in human computation), which could extend the reach of IEC to more complex problems in the future.

2 Background

This section reviews IEC and the module-based ESP method for evolving hierarchically composed neural networks. These are the key ingredients and motivation behind the IEC-ESP method presented in this paper. Evolution of individual modules is based on NEAT [14], which is reviewed next.

2.1 NEAT

Neuroevolution of Augmented Topologies (NEAT) has proven successful in a variety of domains and was originally proposed by Stanley and Miikkulainen [14] as a way to evolve ANNs capable of solving complex control tasks. NEAT is based on three main concepts: (1) NEAT gives a principled way to crossover different topologies; (2) It keeps track of and preserves innovation; (3) It is based on the principle of minimal structure, which ensures that all complexity in a topology is justified. For a more complete review of NEAT see Stanley and Miikkulainen [14].

2.2 ESP Method

ESP is a neuroevolution method to create complex and hierarchically composed behaviours introduced by Lessin et al. [10, 11]. ESP stands for *encapsulation, syllabus and pandemonium*, which are the key ingredients for this method. A *syllabus* in ESP allows complex behaviour to be broken down into more approachable tasks by a human designer. *Encapsulation* allows sub-skills to be preserved throughout further evolution by freezing parts of the ANN. Finally, conflicts between competing modules can be resolved by placing them in a *pandemonium* relationship, in which only the module with highest stimulation will be active, while the rest of the modules will be suppressed.

Task-decomposition strategies similar to ESP have been employed in multiple related field. Brooks's subsumption architecture is a prominent example in artificial intelligence and robotics [15]. In reinforcement learning and evolutionary computation, work such as layered learning and hierarchical task decomposition [16–18] explore similar concepts.

However, a drawback of the original ESP method (and many related approaches) is that the experimenter has to design and fine-tune individual fitness functions for each of the different modules, which can become laborious for complex behaviours. Therefore, the idea in this paper is to evolve those modules through interactive evolution, which is explained next.

2.3 Interactive Evolutionary Computation (IEC)

Typically, IEC frameworks offer the user a set of candidates to evaluate. Users select individuals based on their preference, which are then combined and mutated to produce the next generation of candidates [6]. One advantage of this form of human-based evaluation is its flexibility and expressiveness compared to the explicitly defined mathematical fitness functions traditionally employed in evolutionary computation. For this reason, IEC is used in domains were goals are hard to define mathematically, or were goals might be subjective, such as in creating two-dimensional images [19], three-dimensional forms [20], or musical compositions [21]. Perhaps the most common problem with IEC is user-fatigue; according to Takagi even 10 to 20 iterations may be enough to produce user-fatigue [6], although this depends on the specific details of the task at hand.

A way to alleviate the problem of user fatigue is to combine the intuition of humans with the speed of machines to limit the number of candidates the user has to evaluate [22, 23]. IEC has recently also been combined with novelty search [4], which allows users to focus on evaluating a limited set of *novel* behaviours [8]. A similar set-up was further explored by Löwe and Risi [9] to facilitate the evolution of more adaptive agent behaviours. However, while most related works aim to reduce the number of user evaluations needed, how to create more complex and *hierarchically* composed behaviours has so far received limited attention, which the approach presented in this paper tries to address.

3 Approach: IEC-ESP

The main idea in the presented approach is to allow users to (1) break down complex tasks into smaller sub-tasks, (2) interactively evolve behaviours for these sub-tasks, and (3) interactively evolve regulatory modules that learn when to use each sub-tasks. In this context, the proposed approach allows humans to apply their intuition at two different levels of abstraction. First, as in traditional IEC, humans have the ability to identify promising stepping-stones through the interactive selection process. Second, humans often have a good intuition in how to divide a task into smaller, easier solvable sub-tasks.

3.1 IEC-ESP Algorithm

This section explains the approach in more detail based on a step-by-step example of how IEC-ESP can evolve an agent that can drive through a car track and stop when encountering red traffic lights (Fig. 1).

First, the user decides how to break down the task into smaller components that can be solved using IEC. When a new module is created, the user can also optionally select the input and outputs to which it will be connected. By default, the module will connect to all inputs and outputs, but in some domains it might be useful to limit the available inputs and thereby the evolutionary search space. Determining which inputs and outputs are relevant for a module may be intuitive for a human user, but not directly obvious for automatic approaches.

Individual modules are evolved by NEAT using IEC (Fig. 1.1). During the interactive step users can mark individuals they find interesting or individuals that should be removed. When the selection is complete, users can call for a new generation of individuals, in which candidates that were selected get a higher fitness (10.0 for the experiments in this paper) and therefore have a higher chance to reproduce than unselected candidates (who receive a fitness of 1.0). Basic behaviours can easily be evolved in this way, such as *Drive* (Fig. 1.1) and *Stop* (Fig. 1.2). These basic behaviours can then be orchestrated by regulation modules, which are evolved to decide when their child modules are active (Fig. 1.3). Regulation modules encapsulate everything within them and can be treated as basic modules themselves, allowing for increasingly complex behavioural hierarchies. Alternatively, users can decide on a fixed strategy when modules should be activated, such as "this module will be active/inactive when this [selected] input is active/inactive".

As an advanced feature, users can also directly regulate the contribution of each module to the final result. One module-specific value adjusts all the weights connecting the local outputs of the module to the final global outputs. For most tasks this is not necessary, but it enables fine-grain control over how modules should be combined. For example, by using a negative contribution, it is possible to "subtract" the behaviour of a module (see section 5.1).

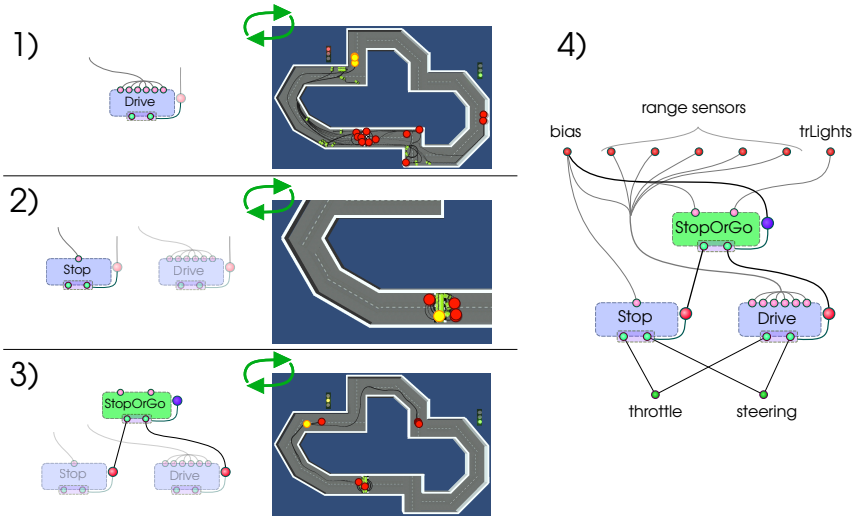

Fig. 1. Incremental Evolution with IEC-ESP. The evolution of the first module that drives regardless of traffic lights is shown in (1). The user can select interesting drivers (yellow) and mark others for deletion (red). The user continues evolving a "stop" module (2), and then a regulation module (3), which decides when *Drive* and *Stop* are active. This module is evolved in the same IEC fashion. A schematic representation of the final neural network is shown in (4).

4 Experiments

We compare the IEC-ESP approach to a regular fitness-based approach and regular IEC without the ability to encapsulate behaviours on three different domains. The first and most simple domain is the well-known XOR logic gate. The second domain is composed of four driving tasks with variable difficulty and the last domain is a garbage collection scenario, which involves sequential and conditional tasks. Based on feedback during the XOR tests, the program was redesigned to make the UI more intuitive.

In addition to an expert user (one of the authors), five users participated in the XOR experiment, and seven in the garbage collection task. To get to know the program, users were shown how to evolve a solution in the car racing domain before they tried the garbage collection task. While the user tests are at this point anecdotal regarding the number of participants and the fact that they did not always have precisely the same conditions, they did demonstrate the potential of the IEC-ESP approach.

4.1 XOR

The goal in the XOR domain is to reproduce the characteristic XOR truth table. Because the tasks is non-linear separable it requires at least one hidden neuron and therefore makes a good initial test case for the modular IEC approach. Before

the test, users were told that they would probably need at least two modules, but were free to try other approaches (which they did). The IEC interface for the XOR problem is shown in Fig. 2. Users evaluate a population of 50 neural networks, whose output is shown in form of a truth table. As a point of reference, users also see the correct truth table (Fig. 2, top), while using the system.

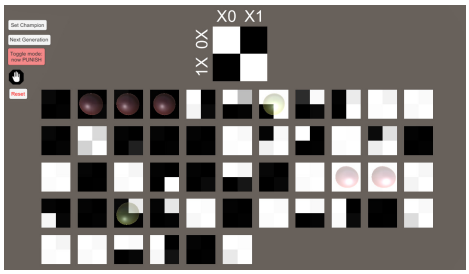

Fig. 2. XOR IEC-ESP Interface. Users are presented with a total of 50 individuals they can choose from (bottom). The truth table for a correct XOR gate is shown on top as reference (1.0 = black, 0.0 = white)

For the fitness-based approach, evaluation is based on: $F = (1 - |t - x|)$ if $|t - x| > 0.5$, and $F = 1.0$ otherwise, where t is the expected output and x is the network's output. The final fitness is the sum over all four input-output pairs.

4.2 Car Navigation

The car domain includes four tracks with increasing levels of difficulty, which should test IEC-ESP ability to create complex and hierarchically composed behaviours. In the first track (Fig. 3a) the only goal is to drive around the track. This racing track is included in the original SharpNEAT port to the Unity platform by Daniel Jallov[1], upon which the system presented here builds. The second track includes traffic lights (Fig. 3b). The two traffic lights are always set to opposite states to reduce the chance of cars encountering two green lights. The third track adds left-right junctions (Fig. 3c), in which junction sensors inform cars of the path they should take (some of these change randomly while the cars are driving through the track). The final track includes both traffic lights and different types of junctions such as T-junctions and dead-ends (Fig. 3d).

Cars are equipped with five short-range proximity sensors that respond to obstacles in different directions (forward, left, right and forward-left and forward-right at 45° angles). Another sensor responds to near-by traffic lights, and is set to 0.5 for orange lights, 1.0 for red lights, and 0.0 otherwise. An additional sensor indicates which direction the car should take at junctions (set to 0.33 for left turns, 0.66 to continue straight and 1.0 for turning right).

For the fitness-based approach $F = [d \cdot c_d - w \cdot c_w - l \cdot c_l] \frac{1}{T}$, where d is the number of road segments advanced, w is the number of collisions with walls and l depends on the behaviour with respect to traffic lights. If the car moves during a green light (or no traffic light is visible) l is decreased one unit, or

[1] https://github.com/lordjesus/UnityNEAT

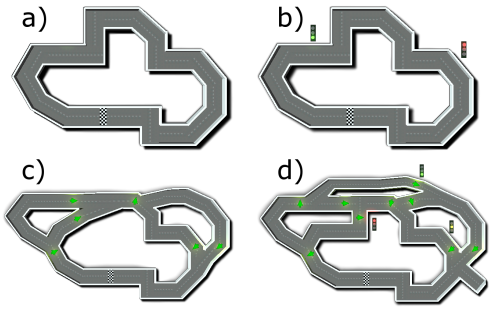

Fig. 3. Car Navigation Tasks. In the simplest track (a) the only goal is to drive around the track as fast as possible. Tracks (b) and (d) include traffic lights. In (c) and (d) green arrows indicate the direction the cars are supposed to follow at junctions.

increased if the vehicle is not moving. The opposite is true for red lights (± 600). The c constants that were found to work best through prior experimentation are ($c_d = 20, c_w = 0.5, c_l = 0.025$). T is the length of the simulation, which was set to a value that would allow good controllers to complete slightly over one lap.

4.3 Garbage Collection

In this task, robots need to transport garbage from two different loading bays to an unload area (Fig. 4). A clock alternates between *red* and *blue* periods, indicating the correct loading bay at that specific time. During IEC the users can change the clock manually, while it changes with a period of 80 time units (experiment length is 150 time units) in the automated setup. After picking up cargo, the robot needs to proceed to the unloading area.

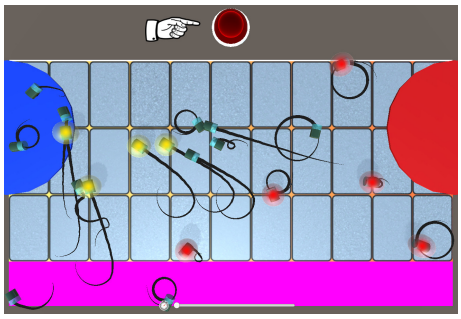

Fig. 4. Garbage Collection Domain. The goal in the garbage collection-domain is to pick up garbage from either the red or blue area (depending on the current clock setting shown at the top), and drop it off at the unloading area at the bottom. The IEC framework allows users to give behaviours either a high (robots shown in yellow) or low fitness (red). For clarity, only 20 individuals are shown in this figure, while 50 are used during the experiments. Black lines show the recent robot paths.

Robots have three range sensors, one pointing straight ahead and two pointing at $\pm 45°$. These sensors only respond to walls and the borders of coloured areas, and return a value between [0.0, 1.0] according to the distance to the obstacle. Their range is limited to about a quarter of the arena length. Three additional long-range sensors provide information about the loading bays and the unloading area. Each sensor responds to one of the three areas and returns

a value that is higher the closer the robot is to the corresponding area. Robots also have a sensor that detects cargo (1.0 for cargo, 0.0 otherwise) and another one for the clock state (1.0 for red period, 0.0 for blue). Robots do not interact (or sense) each other during interactive evolution.

Fitness for the automated approach is set to $F = \overline{s} \cdot c_s - [w \cdot c_w + fit] \frac{1}{T}$, where $\overline{s}$ is the average speed of the robot, and w are the collisions with walls, which is also increased when walls are within the proximity-sensor range, using $1 - \frac{dist}{range}$. The variable fit accumulates values depending on the displayed robot behaviours: $+40$ for picking up cargo, -80 for entering the wrong bay, -20 for entering a bay with cargo, $+50$ for delivering cargo and -10 for entering the delivery area without cargo. Finally, T is the simulation length, and the constants are $(c_s = 0.3, c_w = 0.03)$.

4.4 Parameters

IEC-ESP experiments use a population of 50 individuals classified into 10 species using the k-means algorithm. For the fitness-based simulations we also tested larger population sizes of 150 individuals with 30 species. In the initial population connections from local input to local output neurons are formed with a chance of 99%. 20% of the genomes in each species are copied without changes to the next generation. Parents for the next generation are taken from the fittest 20% individuals in the specie with a chance that is proportional to their fitness value. 20% of the offspring will be produced by asexual reproduction (mutations on a single parent genome). Of the remaining sexual reproduction, $0,1\%$ will be intra-species (parents from different species). In sexual reproduction, common traits are copied from one parent at random. Other traits are always copied from the fittest parent, and all non-common genes from the least fit parent will be copied with 10% chance.

Offspring (except elites) is mutated with a 3% chance of node and 5% chance of connection addition. Connections are removed with $0,4\%$ chance. With 80% probability connection weights are changed. In this case, 99.1% of the time 90% of the connection weights suffer small variations $(\sigma = 0,02)$ while 0.9% of the time 10% of the connections will be reset to a random value within the allowed range from -5 to $+5$. Recurrent connections are allowed. The network is activated five times at each step.

5 Results

The results reported in this section include many trials by the authors, some small-scale user tests and fitness-based simulations. The reported number of generations for the IEC approaches are measured in how many times the user selected the desired candidates and called for a new iteration (offspring creation). A video of the IEC-ESP system in action can be found at: https://goo.gl/VyGE9p.

5.1 XOR Results

In a set of ten independent evolutionary runs, fitness-based search was able to solve the task in an average of 31 generations (sd=18). To solve the task through IEC-ESP, the five testers needed 24, 18, 9, 4 and 16 generations respectively, taking between 30 minutes and one hour (including introduction and set-up). Users reported that becoming familiar with the program was more problematic than solving the challenge itself. After this experiment, the user interface was redesigned thoroughly with the goal to make it more intuitive. At least two users (as well as the expert user) tried to find a solution using traditional IEC; all failed, even after evolving networks for more than 40 generations.

Some different solutions discovered by the users are shown in Fig. 5. Users first interactively evolved two modules with complementary patterns, and then fixed the relative contribution of each one to achieve the desired result.

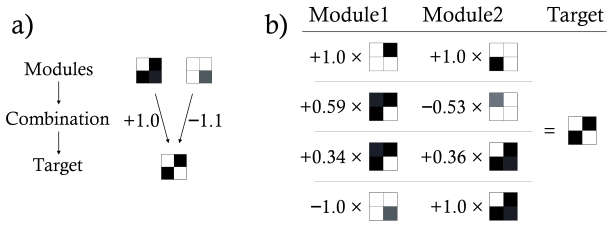

Fig. 5. XOR IEC-ESP Solutions. Schematic decomposition of a XOR gate is shown in (a). Modules are combined with different user-defined weights. Different solutions for the XOR problem are shown in (b). The last solution is from the expert user.

5.2 Garbage Collection Results

While the fitness-based approach can solve XOR consistently, it does not reliably find a solution for the garbage collection task. A variety of different fitness functions were tried, all with limited success.

With the best combination (see Section 4.3), fitness-based evolution solved the task in 2 out of 15 runs. In another three runs (Fig. 6b) one of the two clock periods resulted in a faulty behaviour (typically this involves a wide loop that passes through both loading areas in each cycle, which results in a penalty per loop). In 10 out of 13 runs only partial solutions were discovered, such as the one in Fig. 6a. These partial solutions work well for one of the clock periods and remain idle for the other. Simulations with a larger population (150 vs. 50) perform better, but the trend is the same (Fig. 7).

The expert user was able to solve the garbage collection task in all 20 trials it was tried. Traditional IEC attempts (using only one module) were not successful and, in the best case, resulted in partial solutions similar to the one shown in Fig. 6b. The most common strategy for IEC-ESP solutions (either expert or

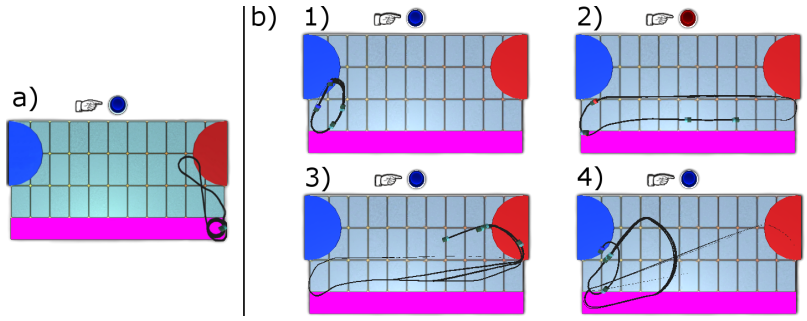

Fig. 6. Typical Partial Solutions for Fitness-based Search. The partial solution in (a) fails when the clock has changed to blue, resulting in a tight-loops idle state. A better approximated solution is shown in (b). The blue loop (1) is almost perfect, while the red unloads too far (2). The red-to-blue transition (3, 4) is problematic for units that were on the way to load. These will go once into the red bay during the blue clock (penalty) and twice into the unloading area without cargo (double penalty).

inexperienced users) was to create three main modules, one to navigate to each of the relevant areas (blue, red and pink). A fourth may be added, since the module navigating to the unloading area usually goes to either the left or right corner of the wide unloading area, which will be inefficient from either the blue or the red bays. This way of approaching the solution results in behaviours that look different from those evolved using a fitness function. A solution found by the expert user is shown in Fig. 8.

The results of the seven non-experienced users were less consistent and they struggled more with the program, which is likely due to the fact that they had to learn how to use the interface on top of solving the task; however, a few solutions (3) were in fact found. For these solutions, each module required in the range of 2 to 50 generations to evolve. Fig. 9 shows such a solution by an inexperienced user that, surprisingly, does not rely on range-finder sensors, resulting in smaller modules. This result demonstrates that even inexperienced users during their first attempts may offer valuable insight for experts, but also shows that getting accustomed to the user interface is key. The main result is that both pure IEC and fitness-based approaches fail when the task requires to elaborate on partial solutions, while IEC-ESP allows especially expert users to interactively and incrementally evolve increasingly complex behaviours.

5.3 Car Navigation Results

The simplest car navigation track (Fig. 3a) is not very challenging for any of the compared approaches. In fact, it is not uncommon to find approximate solutions even from the initial IEC candidates. In the ten independent runs, fitness-based search found a solution in 9.2 generations on average (sd = 9.7; Fig. 10a).

The track with traffic lights (Fig. 3b) proved significantly harder for the fitness-based approach (no perfect solution was found in 30 evolutionary runs,

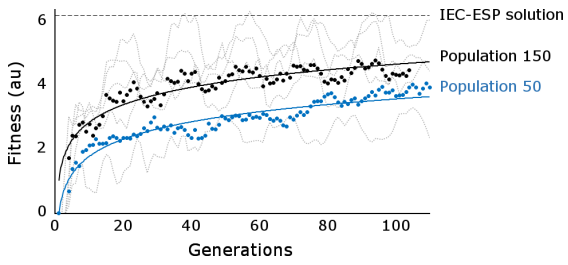

Fig. 7. Average Best Fitness over Generations in the Garbage Collection Task. Light lines are individual simulations (smoothed using 2-generation averages). Full points and continuous lines are average values and logarithmic fit, for experiments with population sizes of 150 and 50. For reference, the average testing performance (sd=0.4) of the IEC-ESP solution (Fig. 9) on 14 trials is also shown (dashed line). Different trials result in slightly different results because of the noisy Unity-based simulation environment.

Fig. 10b). Tracks with junctions proved even more challenging in our fitness-based attempts, so we chose to focus on the easier task with only traffic lights.

The expert user was able to find a IEC-ESP solution for the traffic light track (Fig. 3b) in all 20 attempted trials. Fig. 1 shows an IEC-ESP solution, which is composed of one module that drives with no regard to the traffic lights (Fig. 1.1) and one that allows cars to stop. Notice this is a trivial behaviour, because it needs not decide *when* to stop, but just *how* to stop. For stopping (basically doing nothing) only bias is used as input. The last element is a regulation module that decides whether to drive or stop depending on the value of incoming traffic lights.

A possible strategy to solve the more complex track with junctions (Fig. 3c) is to first evolve two simple driving modules that tend to follow right and left walls. A regulation module is evolved next, which allows the car to switch between the two modules when entering junctions.

Fig. 11 shows two schematic solutions for the track with traffic lights and junctions by an expert user (Fig. 3d). One of the solutions includes a regulation module that decides among *Left*, *Straight* and *Right* (where normal driving is *Left* or *Right* which drives closer to left or right walls respectively). The straight-driving module is necessary for certain T-junctions in this track. This module scheme seems like a good idea, but in practice regulatory modules with more than two children can be challenging to evolve. Although it looks more convoluted, it is actually easier to develop a solution that uses a two-step regulation process. In the second example in Fig. 11, the first module decides between *Left* and other driving modules. If Left is not chosen, then a second regulatory module chooses between *Straight* and *Right*. This approach is easy to scale.

6 Discussion

This paper presents an approach that combines IEC with the ESP method [10, 11], which allows evolved skills to be protected and reused as part of a larger

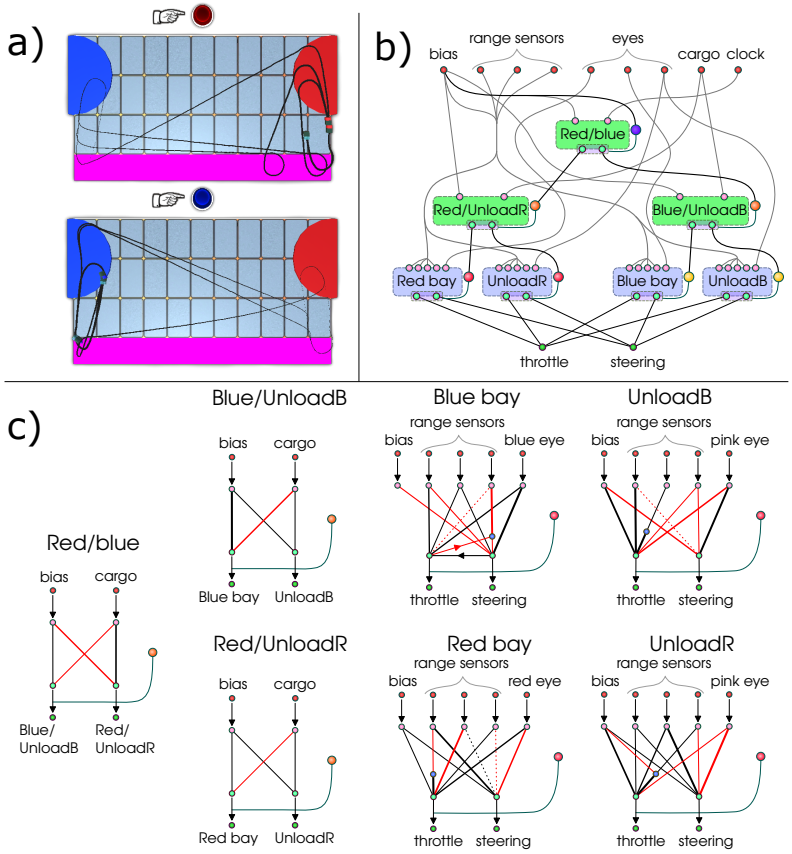

Fig. 8. Expert User Solution for the Garbage collection Task. (a) Behaviour for both periods with trails after clock transitions. (b) Network diagram. Regulatory neurons are at the right side of each module, and share colour when they are in a "pandemonium" (only one in a group may be active at any time). (c) Network connections within modules. Red/UnloadR connections are shown twice as intense for clarity.

network. The results suggest that the approach facilitates the evolution of hierarchically composed behaviours, which is difficult with the traditional form of IEC. The presented method takes a step towards exploiting a larger part of our human intuition in the IEC process, by allowing users to break down the given task into intermediate and therefore easier to solve sub-tasks. Deciding on the inherent problem structure of a task requires high-level abstract reasoning, a problem where humans currently still surpass machines. In the future it will be interesting to use the computational speed of the machine to a fuller extend. For example, a novelty-assisted IEC approach [8] could continuously generate novel behaviours to accelerate the IEC aspect of our IEC-ESP approach.

Comparing the different approaches, it is interesting to note that humans and algorithms have a very different concept of difficulty. The traffic light task is more

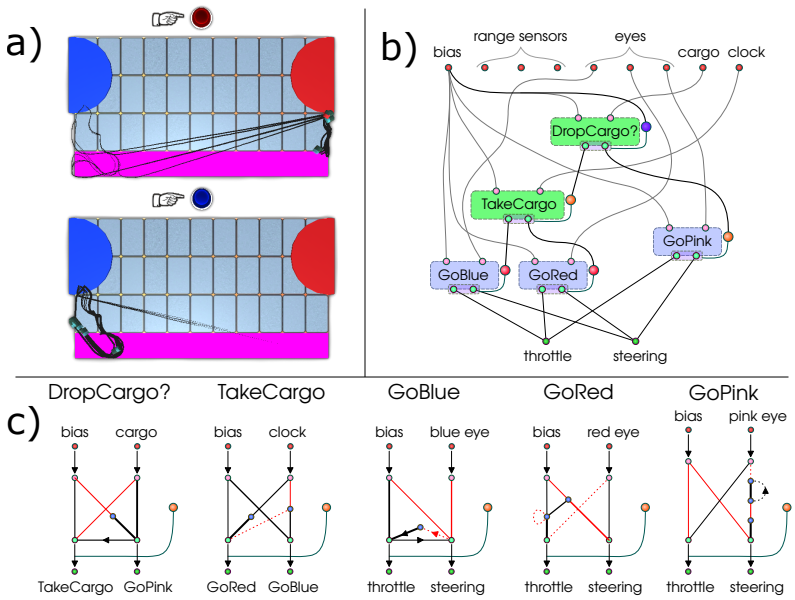

Fig. 9. Non-expert User Solution for the Garbage Collection Task. (a) Correct behaviour for red and blue clock periods. Trails show the transition from the other period. (b) Schematic network view. Regulatory neurons are represented at the right side of each module. Regulatory neurons of the same colour share a "pandemonium", allowing only one in the group to be active at the same time. (c) Network connections within modules. Note this solution does not use range sensors at all.

challenging for the automated approach than the garbage collection problem, while the reverse is true for humans. This difference does hint at the potential benefits of learning how to combine human and computer intelligence efficiently. In particular, we believe the failure in the driving task is more related to the unpredictable environment and noisy evaluations rather than the complexity of the behaviours involved. This is problematic for the fitness-based approach, but not for IEC-ESP. While a different fitness function, incremental evolution or a more deterministic environment could alleviate these problems, it shows the versatility of the presented approach that succeeds without any such restrictions.

Although the results are promising, some IEC-ESP solutions for the most complex track have small shortcomings. While the best solutions can perform several laps without any mistakes, sometimes transitions between different modules can be problematic and result in unexpected behaviours (e.g. loops or driving in the wrong direction). Mitigating problems that can result from the combination of apparently successful modules is an important future research direction.

An interesting insight from this work is the importance of training the users to use the particular interface and getting to know the program and domain. As the results show, while an expert user with intimate knowledge of the system and underlying algorithms can consistently solve the tasks, the same is not true for

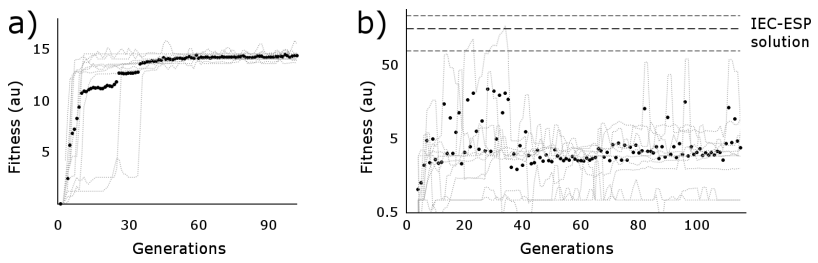

Fig. 10. Average Performance. Fitness over generations for the tracks without (a) and with traffic lights (b). Black dots show averaged fitness over ten independent evolutionary runs. Dashed lines in (b) show the average test performance of a IEC-ESP solution evaluated on 40 trials together with ± sd.

the non-expert users. In the future it will be important to alleviate this problem by making the user interface more intuitive but also providing a step-by-step and easy to understand tutorial.

One exciting promise of the approach is that the modular-network structure makes it possible for multiple users to collaborate on their design. In case of a monolithic neural networks, reusing parts of a network for a new problems would be significantly more challenging. For example, an interactively evolved module that processes information from ten light-sensitive neurons could be used in any project that implements such sensors, without the rest of the network interfering.

The experiments reported here are preliminary and unfortunately do not allow for a proper statistical analysis. Therefore, important future work is to apply IEC-ESP to relevant problems and to conduct rigorous and extensive user-experience experiments.

7 Conclusions

This paper introduced a combination of IEC and the module-based ESP method for preserving, combining and reusing previously learned skills. The new IEC-ESP method was evaluated on a variety of different tasks that proved challenging to evolve with a traditional fitness-based approach. IEC-ESP requires the experimenter to decompose the overall task into smaller sub-tasks, which fits naturally with our human abilities of high-level abstraction. Importantly, the defined sub-tasks can then be evolved efficiently through interactive evolution. The main conclusion is that the approach now allows more complex domains to be solved than have heretofore been possible with IEC, and adds more evidence to the synergistic effects in combining human and machine capabilities.

8 Acknowledgements

We thank Fundación Ramón Areces for funding as part of their postdoc fellowship program.

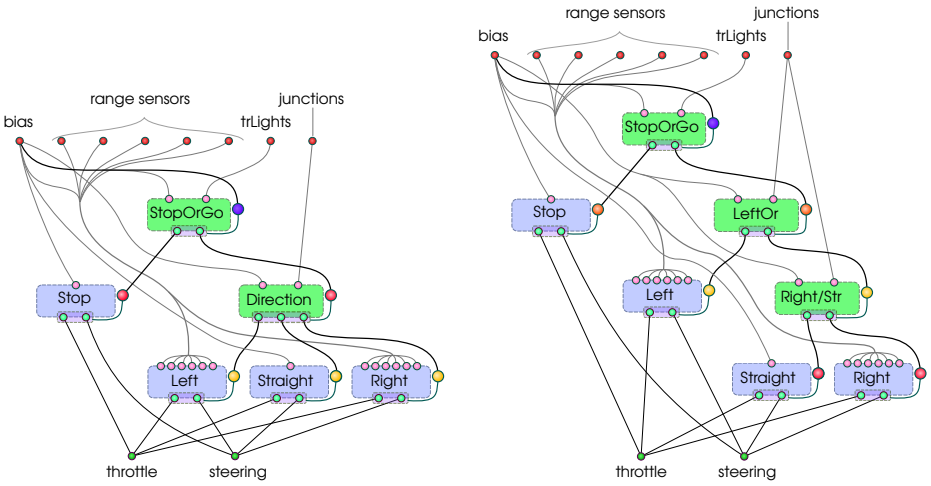

Fig. 11. Two Solutions for Car Track d. *Left*, *Straight* and *Right* driving behaviours may be controlled by one regulatory module (compact, harder to evolve) or by a nested sequence of simple regulatory modules (more complex hierarchy, but easier to evolve).

References

1. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence **1**(1) (2008) 47–62
2. Yao, X.: Evolving artificial neural networks. Proc. of the IEEE **87**(9) (1999) 1423–1447
3. Risi, S., Togelius, J.: Neuroevolution in games: State of the art and open challenges. IEEE Trans. on Computational Intelligence and AI in Games **PP**(99) (2015) 1–1
4. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. In: Proceedings of the Eleventh International Conference on Artificial Life (Alife XI, MIT Press (2008)
5. Goldberg, D.E.: Simple genetic algorithms and the minimal, deceptive problem. Genetic algorithms and simulated annealing **74** (1987) 88
6. Takagi, H.: Interactive evolutionary computation: Fusion of the capacities of ec optimization and human evaluation. Volume 89. (2001) 1275–1296
7. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation **19**(2) (2011) 189–223
8. Woolley, B.G., Stanley, K.O.: A novel human-computer collaboration: Combining novelty search with interactive evolution. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. GECCO '14, New York, NY, USA, ACM (2014) 233–240
9. Löwe, M., Risi, S.: Accelerating the evolution of cognitive behaviors through human-computer collaboration. In: Proc. of the Genetic and Evolutionary Computation Conference 2016. GECCO '16, New York, NY, USA, ACM (2016) 133–140
10. Lessin, D., Fussell, D., Miikkulainen, R.: Open-ended behavioral complexity for evolved virtual creatures. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. GECCO '13, New York, NY, USA, ACM (2013) 335–342

11. Lessin, D., Fussell, D., Miikkulainen, R., Risi, S.: Increasing behavioral complexity for evolved virtual creatures with the ESP method. arXiv preprint arXiv:1510.07957 (2015)
12. Michelucci, P., Dickinson, J.L.: The power of crowds. Science **351**(6268) (2016) 32–33
13. Khatib, F., Dimaio, F., Cooper, S., Kazmierczyk, M., Gilski, M., Krzywda, S., Zabranska, H., Pichova, I., Thompson, J., Popović, Z., Jaskolski, M., Baker, D.: Crystal structure of a monomeric retroviral protease solved by protein folding game players. Nature Structural and Molecular Biology **18**(10) (10 2010) 1175–1177
14. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2) (2002) 99–127
15. Brooks, R.: A robust layered control system for a mobile robot. IEEE journal on robotics and automation **2**(1) (1986) 14–23
16. Doucette, J.A., Lichodzijewski, P., Heywood, M.I.: Hierarchical task decomposition through symbiosis in reinforcement learning. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation, ACM (2012) 97–104
17. Whiteson, S., Kohl, N., Miikkulainen, R., Stone, P.: Evolving keepaway soccer players through task decomposition. In: Genetic and Evolutionary Computation Conference, Springer (2003) 356–368
18. Lee, W.P., Hallam, J., Lund, H.H.: Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In: Evolutionary Computation, 1997., IEEE International Conference on. (Apr 1997) 501–506
19. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: A case study in collaborative evolutionary exploration of design space. Evol. Comput. **19**(3) (September 2011) 373–403
20. Clune, J., Lipson, H.: Evolving 3d objects with a generative encoding inspired by developmental biology. SIGEVOlution **5**(4) (November 2011) 2–12
21. Hoover, A.K., Szerlip, P.A., Norton, M.E., Brindle, T.A., Merritt, Z., Stanley, K.O.: Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In: Proceedings of the Third International Conference on Computational Creativity, Dublin, Ireland (may 2012) 111–118
22. Bernatskiy, A., Hornby, G., Bongard, J.: Improving robot behavior optimization by combining user preferences. In: ALIFE 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems. (2014)
23. Wagy, M.D., Bongard, J.C.: Combining computational and social effort for collaborative problem solving. PloS one **10**(11) (2015) e0142524